Six Degrees

ex: Carrie Mary Caleb
    Mary Jim Carrie
    Jim Michael Mary Steve
    Michael Caleb Jim
    Caleb Carrie Michael
    Steve Jim
    Donald

2 goals:
(1) Avg degree of separation.
(2) Query

Query: A = Mary / B = Caleb
Want: Mary -> Carrie -> Caleb
Not: Mary -> Jim -> Michael -> Caleb

Want shortest chain!

will use an algorithm to find chain
Algorithm requires a Person class:

Step 1: Create Person class:
Data variables:
- name (String) (parameter of constructor)
- friendList - list of names of friends (ArrayList of Strings)

one other explorer - whether this person has been searched (boolean)
things joined - name of previous person in chain (String).
Step 2: Driver:

1. Create list of Persons using input file.

```java
import java.io.*;
public class Driver {

    static ArrayList<Person> peeps; // global

    public static void readFile(String file) throws IOException {
        Scanner fr = new Scanner(new File(file));

        while (fr.hasNextLine()) {
            String line = fr.nextLine();
            String[] names = line.split(" ");
            // read first name to create Person
            Person person = new Person(names[0]);

            // read rest of names for friendList
            for (i = 1; i < names.length, i++)
                person.friendList.add(names[i]);

            // add this Person to peeps
            peeps.add(person);
        }
    }
}
```
**Step 3**: Run Search Algorithm

A = Mary  B = Caleb

**Visualize**

```
  Mary
   / \  \\
  Jim  Carrie
 / \    /  \\
Mary Michael Steve Caleb Mary
```

Algorithm starts with Person A (Mary), explores friends, then friend's friends, etc. until Person B is found.

Don't want to explore the same person multiple times so keep track of whether a person has already been explored in Person class (boolean explored)

Need to keep track of "chain" in Person class: String predecessor
Carrie, Mary, Caleb
Mary, Jim, Carrie
Jim, Michael, Mary, Steve
Michael, Caleb, Jim
Caleb, Carrie, Michael
Steve, Jim

Search (Mary, Caleb) \[A = \text{Mary}, B = \text{Caleb}\]
\[\exp = T\]

Explore List

\[
\begin{array}{cccccc}
\text{Mary} & \text{Jim} & \text{Carrie} & \text{Michael} & \text{Steve} & \text{Caleb} \\
T & T & T & T & T & T \\
\end{array}
\]

pred
null, Mary, Mary, Jim, Jim, Carrie

\[X = \text{Mary, Jim, Carrie, Michael, Steve, Caleb}\]
\[Y = \text{Jim, Carrie (back to while), Michael (don't add Mary for Jim), Steve (back to while), Caleb (don't add Caleb, Jim for Michael), Jim for Steve)}\]

Follow pred's:

Caleb \leftarrow Carrie \leftarrow Mary

\[\text{Start with Person A!}\]
Print chain:  A = Mary  B = Caleb

current:  Caleb  Carrie  Mary  null

ChainList:

| Caleb | Carrie | Mary |

ChainList.size() - 1 = # degrees of separation between A and B.